Aalto University
School of Science
Master's Programme in ICT Innovation

Brendan Goubin

Collaborative Indoor Mapping in a Spatial Data Management Context

Master's Thesis
Espoo, July 9, 2015

Supervisors: Professor Jukka Nurminen
Advisor: Mikko Virkkilä

Aalto University
School of Science
Master's Programme in ICT Innovation

ABSTRACT OF
MASTER'S THESIS

Author: Brendan Goubin

Title:
Collaborative Indoor Mapping in a Spatial Data Management Context

Date: July 9, 2015 **Pages:** 51

Supervisors: Professor Jukka Nurminen
Advisor: Mikko Virkkilä

The objective of this thesis was to research on collaborative geospatial data management, how to share, store and process data between different actors, of different natures and different purposes. This research work will serve as base for the implementation of a web-based collaborative map editing application for the company Nimble Devices Oy, and to be partly published as a free licensed software.

The geospatial data is being stored as generic geometric objects in a PostgreSQL database. Every data processing will be represented as an "Action" object containing all the required informations to replicate the same data processing on any other actor of the whole system. The "Action" objects are transmitted to the database through an application programming interface on the server side, and will be transmitted to the other web clients with the Javascript library TogetherJS.

This thesis will present background research and a literature review on the topics of geospatial data and indoor mapping. Then a specification and conception work will be done to suggest solutions for the previous problematic. The thesis then presents some technical validation and results of the project. And finally, a simple conclusion including the future works, and future possibilities related to this work.

Keywords: Spatial Databases, Data Management, Collaborative Work, Indoor Mapping

Language: English

Acknowledgements

I want to thank Jukka Nurminen, for being my supervisor, and for the help he provided me concerning the master's thesis. I would like to express my gratitude to Mikko Virkkilä, for offering my the opportunity to work on my master's thesis as an intern at Nimble Devices Oy. I also want to thank the whole Nimble Devices Oy team: Tuomas Ilola, Sam Pullen and Ruud Visser, for the working environment.

Espoo, July 9, 2015

Brendan Goubin

Abbreviations and Acronyms

OGC	Open Geospatial Consortium
OpenGIS	Open Geographic Information System
CAD	Computer Aided Design
NDD	Nimble Devices Data
DXF	Drawing eXchange Format
JSON	JavaScript Object Notation
HTML	HyperText Markup Language
CSS	Cascading Style Sheet
GPS	Global Positioning System
Rest	REpresentational State Transfer
API	Application Programming Interface
SDMS	Spatial Data Management System
BIM	Building Information Modeling
GIS	Geographic Information System
UUID	Universally Unique IDentifier
RSSI 1M	Received Signal Strength Indication : 1 meter
GPL	General Public License
AGPL	Affero General Public License
BSD	Berkeley Software Distribution License
MIT	Massachusetts Institute of Technology License
MPL	Mozilla Public License

Contents

1 Introduction

1.1 Motivation

During this thesis, we will study data management and conceive a solution to create geospatial data, to edit it, to display it and to store it. This solution will include collaborativity features.

To support the theoretical work, a technical experimentation will be made, and we will implement a map editor for the company Nimble Devices, to be used within their indoor positioning solution, during the digital representation of indoor environment process.

1.2 Nimble Devices

Nimble Devices is a Finnish company specialized in indoor positioning and navigation, and more precisely in mobile indoor positioning. The company is developing an indoor positioning solution, and then integrates it in different mobile applications, for different clients. Thanks to the wireless technologies, it is possible to localize a device with good accuracy in an indoor environment and with this, Nimble Devices develops a solution that provides routing and guiding assistance, they can also include real-time information depending on the positioning of the user, and the solution can also trigger events when the device of a user enters a specific room for instance.

1.3 Research Questions

During this thesis, we will consider different problems, some specific to the map creation and the map editing, some for the geospatial data management, and others for handling a collaborative service.

We will need to figure out how to store the data of the different maps, how the different information, or the different objects composing the maps

will need to be organized.

How to handle a map creation or edition, with as much basic data as possible in order to expand the different use cases, and have a generic tool that can offer more possibilities than a simple map editor.

And finally how to handle the collaborative part, how the data should be shared between the different actors, how will it be replicated, should the resource be locked or should a limited part of the resource be locked when an edition occurs.

Additionally, one of the purpose of this work was to simplify a task of the map creation process for the company Nimble Devices, as explained in the following chapter 2, the digital representation of an indoor environment takes time and requires training to have a full control over the used software. Therefore, the solution will be conceived and implemented in the perspective of offering a tool that is easy to use, easy to understand, and that will not require as much time to be mastered.

1.4 Collaborative Application

Nowadays team work is a huge part of the working life. The developed solution will include a collaboration part in order to improve the user experience. The first focus of this collaborative part will be to have a simple monitoring feature. Allowing a user to actually use the software, and some other users to monitor its work. This would be useful for pair working for instance. The later focus will be to develop a collaborative part that would allow two users to use and share the same data, and to work on the same project without interfering with each other.

In a more technical focus, different tools, libraries and framework exist for providing the different collaborative features to a JavaScript application: from basic mouse clicks sharing to more complicated data live transfers. Therefore, we will study and compare different solutions for collaborative handling, and choose the most appropriate one.

1.5 Indoor Mapping and Indoor Positioning

Indoor mapping is the digital representation of indoor environments. It is a tool relevant in many situations, and use cases. Indoor positioning is the process of defining the position of an object in an indoor environment, it can be done with different technologies with different accuracy using different methods. [24] [32] [38]

Indoor mapping and positioning can be done with different methods and technologies, such as Bluetooth, Wifi or Global Positioning System for instance. The chosen technology depends on several criteria, the same technologies cannot necessarily be used for indoor environment and outdoor environment, the accuracy requirements may not provided by every solutions.

1.6 Spatial Data Management

Spatial Data Management is the storing of spatial data in specialized databases, most of all treating these data as specific spatial data instead of simply treating it as generic data. This involves specialized and optimized storage and retrieving of data.

The Open Geospatial Consortium defined specifications and standards for implementing database management systems with geospatial data treatment.

For instance, the kind of specific geospatial data management operations can be querying data within a certain geographical area.

Query	**Input**	**Output**
Distance	geometry,geometry	number
Equals	geometry,geometry	boolean
Disjoint	geometry,geometry	boolean
Intersects	geometry,geometry	boolean
Touches	geometry,geometry	boolean
Crosses	geometry,geometry	boolean
Overlaps	geometry,geometry	boolean
Contains	geometry,geometry	boolean
Length	geometry	number
Area	geometry	number
Centroid	geometry	geometry

Table 1.1: Typical geospatial query types.

1.7 Structure of Thesis

The structure of this thesis is as follows: In the chapter 2, we discuss about the backgrounds of the thesis, and the technical project and presenting the related work.

Then in the chapter 3, we explain the conception and modeling of the solution, the chapter contains the organization of the solution and the technical choices.

Chapter 4 will describe the implementation of the technical work, it will detail the general organization of the project, how the collaborative features are handled.

After implementation, the chapter 5 will be defining the validation process of the project and will present some results.

In the continuity of the validation part, a chapter 6 will focus on the usability of the implemented solution.

Finally, in the last chapter 7, we will summarize the thesis, and discuss about the next steps relative to the topic and technical project.

2 Background and Related Work

In order to handle indoor data in a spatial context, different existing technologies and approaches are relevant, they can provide clues and insights for structuring a solution.

2.1 Spatial Data Management

Spatial data is data that includes a geographical positioning context. Whether it is a moving object or a fixed positioned item. Spatial Data Management is a technique for organizing and retrieving information by positioning it in a spatial framework. It is then accessed with a Spatial Data Management System (SDMS). [19]

2.1.1 Spatial Data Management System

Different ways of handling data have been used and tried, expressing queries in natural language, giving example of the desired results or graphically describing the retrieval process. But those approaches still require that the user precisely specify the data that will be retrieved, which means that the solution requires to know what the database contains, and how it is organized. [19]

Therefor, to solve this issue, Spatial Data Management Systems organizes the different information in two-dimensional geometric spaces. This technique is more intuitive for untrained users.

Building such a system requires the resolution of two problems: [19]

- A mean that will be used to view a large data surface.
- A mechanism for creating pictorial representations of symbolic information.

2.1.2 Spatial Data Structures

Spatial data consists of spatial objects composed by points, lines, regions, rectangles, surfaces, volumes, or possibly data of higher dimensions. A way to handle it is to store the data and parameterize it and then obtaining a reduction to a point in a possibly higher dimensional space. [34]

Also, the data requested needs to be retrieved with queries that are based on not explicit spatial properties. This involves a possible high volume of data, and a high processing cost, while computing it on the fly may be more interesting. Even more if the data is stored in an appropriate manner. The spatial data is represented by separating it structurally from the non spatial data and keeping appropriate links between the two types of data. [34]

2.2 Spatio-Temporal Databases

Spatio-temporal databases are database system management that can handle data in both space and time context, this can be used to track objects for instance, defining the history of positions of a tracked object. In a more concrete perspective, those kind of database are used to keep records of history, to understand some scientific questions, or to solve puzzles: understanding the trajectory of bullets, determining migration patterns of animals, when did the president meet the prime minister. [16]

Our implementation will include the geographic position of geometric components, and in the end, the project should handle a history of each objects, to do this, we are going to use this kind of database.

2.3 Computer Aided Design

Computer-aided design can be defined as the use of an automated system to assist in the process of development, edition, analysis or improvement of a design. The created images are composed of basic geometric elements, as dots, lines, circles, etc. [35]

CAD can have the following applications:

- Stress-strain analysis of components
- Dynamic response of mechanisms
- Heat Transfer calculation
- Numerical control part programming

CAD systems increase the productivity of the designers, the quality of its work, the communications through documentation is also improved. [35]

The current system for creating and editing maps is handled with CAD software, the main issue with those kind of programs is that it requires to be trained to use them, these are complex software, that need time and practice to be mastered. The implemented solution aims at removing this barrier, and to make it easy for a first time user to create and to edit its own maps.

2.3.1 Building Information Modeling

The concept of Building Information Modeling (BIM) has been created in the 1970s, it has been imagined when computer technologies were getting more important and would start being interesting for checking processes in the building industry, such as verifying the reliance of drawings for instance. [14]

BIM represents the process development, and the use of a computer generated model to simulate the planning, the design, the construction and the operation of facility. [8]

It is used to visualize what is to be built in simulated environment. Architects, engineers and constructors will use it to identify potential problems in the design, construction or operation process.

The model of information has the following properties is a data rich, object oriented, intelligent and parametric digital representation of the facility.

It carries all information about the buildings, its physical and functional characteristics, the project life cycle information too. For instance, a building information can contain geometry and geographic information, spatial relationships, quantities and properties of building elements, cost estimates, schedule, etc. [25]

BIM is used for different purposes:

- Visualization
- Fabrication
- Code reviews
- Forensic analysis
- Facilities management
- Cost estimating
- Construction sequencing

- Collisions detection

The key benefit of using BIM over some other system is its accuracy concerning the geometrical representations of the different parts of a building. But as it is an earlier system, is has a better use of the available computational power, and benefits from the new technologies aswell. Therefor, it has faster and more effective processes, it generates better designs. You can control the whole life costs and the environmental data. The production quality is improved such as the customer service. [8]

2.4 Geographic Information System

There are different definition of what a Geographic Information System (GIS) is, each one having a different approach.

But the most general definition developed and agreed by specialists is the following: A system of hardware, software, data, people, organizations and institutional arrangements for collecting, storing, analyzing and disseminating information about areas of the earth. [11]

The purpose of a GIS could involve a complex decision or a routine decision, as the policy for timber harvest, or the granting of a permit for instance. It is a tool, that can be designed to be effective and efficient for a certain purpose. [15]

The implemented project will need to handle most of the operations that a GIS performs, in order to provide a functional and standard service.

2.5 Location Model

A location model, is the description of an area where we want to set up a positioning system. This description can be either geometric, which is based on the coordinates of the system for precisely define the position of an object. It can be symbolic, which is a clear description of the relationships between the objects. Finally, the modeling can also combine the two ways at the same time, as a hybrid system, that will show the hierarchical relationship between the objects, and allow a precise determination of their positions. [27]

In our case, location models can be used for indoor location, to handle different types of functions, as tracking objects or users, guiding users through the environment, or defining the state of an entity, for instance.

2.6 Indoor Positioning Technologies

The indoor positioning is a process mostly done with fingerprinting, but can also be done with trilateration, triangulation or image processing:

Fingerprinting is the observation of the characteristics of the different signals received of a device. It is done in two steps, the calibration of the environment, establishing a database of measurements of the different received signals, in different locations. And after that, an estimation of the current position made by comparing the current measurement with the reference locations. [22]

Triangulation is a technique to find the location of an object by measuring the angles between this object's position and other references points with a known position. The object's position becomes the third vertex of a triangle with known angles and a known edge size. [18]

Trilateration is a mathematical method that is used to determine the relative location of an object using the triangle geometry as for triangulation, but this one uses the angles and the distances to process the position of the object. Trilateration uses the distances between at least two reference points. [40]

Image Processing is a technique that uses a camera to capture the view of the user and process the position out of it. The main issue is that this technique requires a high computation power. [33]

Two properties are needed in order to ensure that the positioning is working, the spatial variability, which is the fact that a device will receive very different signal strengths if it moves, considerable changes should be detected for small distances. The second requirement is the temporal consistency, it is needed that the received signals stay the same for a given location. If a device should receive any signal with the same strength at any time. Those two properties ensure the functioning of the fingerprinting technique. [9] [20] [21] [23]

2.7 JavaScript Object Notation

JavaScript Object Notation (JSON), is a text format used to store and structure data. It is derived from the JavaScript object notation. A JSON object can include basic types values, other JSON objects, or arrays of primitive

values, or arrays of JSON objects. The basic types are strings, booleans, or numbers. The JSON format is easily read by machines and by humans and it is easy to learn, because the syntax is highly simplified. [13]

As a JSON object is highly hierarchised, it is easy to represent a map and its inner generic items as JSON objects themselves.

This kind of structure makes it easy to represent hierarchised objects, such as maps, containing different levels, themselves containing different areas and items.

2.8 Application Programming Interfaces

There are different reason for implementing and using programming interfaces:

Security As previously described by the figure 4.1.1, the database is not directly controlled by the clients that would require the credentials. The intermediate service is placed between the database and the clients.

Abstraction The APIs are also used for abstracting the database layer, the clients do not have to handle their database queries. This allows the developer to change the characteristics of the system, switching the database management system for instance, editing only the functioning of the APIs of the system.

Unique behavior An application might be used in different ways, web-embedded in the current case, but could be used in a mobile or a desktop application. Using APIs prevents the developers from writing the same behavior several time on those different devices, or different languages, the only behavior to implement is the calling of the APIs for each platforms.

As the application will be developed for a web-based solution, we do not use the APIs for the unique behavior advantage, but we are using those for the security issues and to have an abstraction layer between the client and the database.

2.9 Collaborative features

Nowadays, the ongoing trend is to develop real time collaborative application, allowing several users to use the same application or to edit the same document for instance, increases the efficiency of the team, and make it easier to have a good coordination between the members of a group.

2.10 Licensing and re-use of existing technologies

Some part of the work done during this thesis will be published as free licensed software such as Berkeley System Distribution License, or GNU General Public License. So in a general way, the tools we are going to use to implement the solution will have to be free as well. As the work done might be used later by other persons, the tools should also be standardized. It is not a compulsory requirement, but in this way, it would be easier to use or to edit the application.

3 Conception and Modeling

3.1 Overall solution

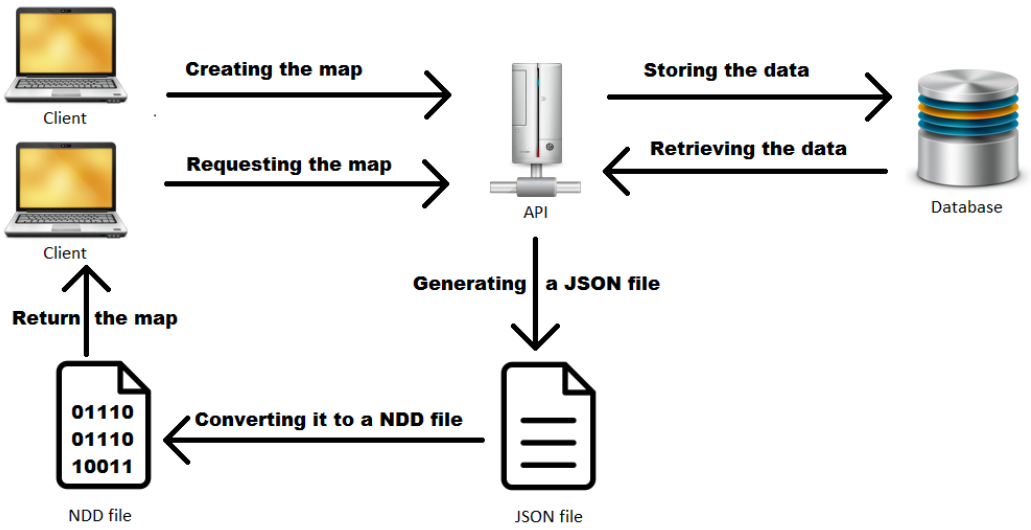

Figure 3.1: The planned architecture of the application.

As shown by Figure 3.1, the application includes a server that will interact with the clients, and a database that will store the information. We can see two main use cases:

Creation and edition of a map: The client will use a web application, synchronized with the potential other clients, to graphically create a map, and handle the different information and entities inside it. Those information will be stored in a database.

Retrieving a map: Once a map is created, a client will be able to request it to the RestAPI. The server will retrieve the data stored in the database,

and generate a JSON file out of it. This JSON file will be processed into a NDD file that will be returned to the client.

3.2 Data Management

As the designed system will eventually replace the old one, it is important to know how the data is stored. Currently, the .DXF file generated by CAD software is converted into a .JSON file, before being converted into a .NDD file.

So the first step before designing the data management, will be to understand how it is currently functioning, and how the maps information are handled and organized.

Then, as we need to handle shared data, it needs to be well organized, and the data sent from a client to the other one should be as restricted as possible to avoid latency between editing and displaying. If a wall is added in the map, it would make more sense and would be more efficient to send the information that a set of points was added in the map, and the information of each points, instead of sending all the map information.

Those ways of handling the data would not make a lot of difference for small maps, but if there is a lot of data to handle, there might be latency and other sort of issues for sending the data, but also for rendering it.

As we want to avoid unnecessary computation, we need to use a simple database, that provides easy access to the data and easy ways of editing this data. The database solution choosing process is discussed in the following part 3.2.2

3.2.1 Old JSON Organization

Map As described by the Figure 3.2.1, a map is identified by a versioning number, can include some tags, contains an object defining the general boundaries of the environment, and finally several levels.

- **Bounds** The bounds of a map, or of a level, basically contains the minimum and maximum limits of it, in x and y coordinates, but also in GPS coordinates with the longitude and the latitude.
- **Level** Each level includes all the required data to build it, it includes the projection data, the different zones, the walls information, the list of beacons of this level, the keepout zones and keepout holes and finally the list of junctions. It also includes the bounds of the level.

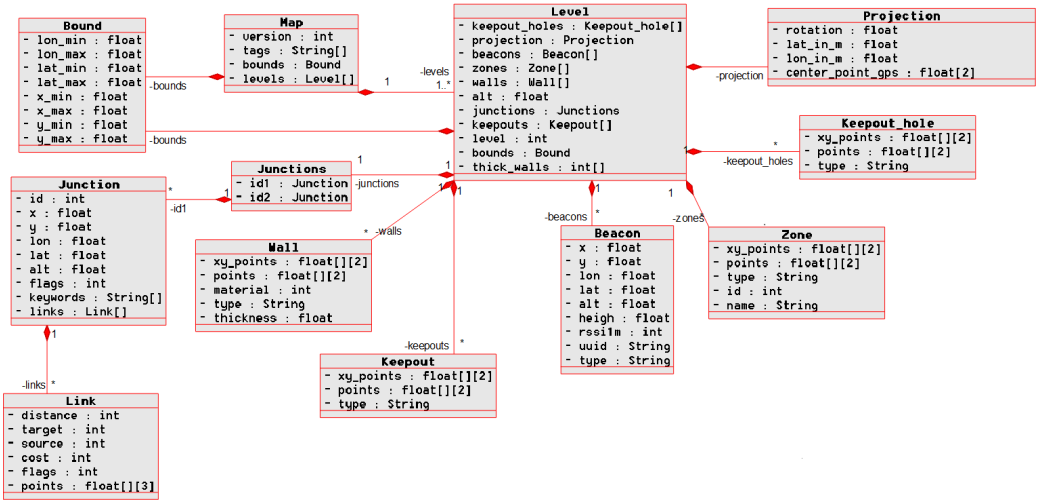

Figure 3.2: JSON Architecture of a map.

Projection The projection data is the required information to process the rotation between the x and y coordinates and the GPS coordinates, it is also used to calculate the distance matching : meter - latitude - longitude.

Zone A zone includes the x and y coordinates of its points, their GPS coordinates as well, the type of zone, and identifier and a name.

Keepout Theses are the zones that you can not be in, it includes the different points of the zone, and the type of the zone. The difference between a keepout and a zone, is that the keepouts do not include any identifier nor any node.

Keepout hole A keepout hole is an allowed zone inside a keepout zone, it has the same information than a keepout zone.

Beacon A beacon is the device that makes it possible to track a user, it has a defined x and y position, it also has GPS coordinates, the altitude and height information. Then for the non positioning data, it includes the UUID, which is the identifier of the beacon, the type of the object, and the RSSI, which is the signal strength.

Wall The walls are defined by the positions of their points, the material that constitutes them, the type of object and the thickness.

Junction The junctions are the vertexes of the defined routes.

They include x and y, and GPS positioning information, flags that are not currently used, they also contains keywords and the related links. The junctions are currently stored in one object called "junctions" as direct objects of it.

Link The links are the edges of the routes, they contains information about the distance, the target and source junctions, the cost of using this edge, for calculating routes. As the junctions, the links contain flags, that are not used either, and the coordinates of it.

3.2.2 Database Choosing Process

As detailed in the annexe A, a list of possible solutions with their main information has been selected: Is the solution a native database, or is it an extension working on top of a database, is the license proprietary or free (GPL, AGPL, BSD, MIT, MPL or Apache license), then are the sources open, and how is the implementation from a standardization point of view, if a solution is openGIS compliant, it means that the Open Geospatial Consortium certifies that the solution is matching the defined openGIS standards. However a solution can be implemented in order to match the openGIS specifications, without being validated and certified by the OGC.

The important points for choosing a database over the others will be using an open solution, in terms of licensing, but also in terms of standardization, as explained in the section 2.10. So the first filtering will be to remove the proprietary licensed databases. Then we select only the databases that are implemented in order to match the openGIS standards, and the databases that are openGIS compliant. Then a major filtering will be done according to the popularity and stability of solutions, in order to have a better documentation, and a better eventual support during the development of the final solution.

The final possibilities that could be used would be MySQL , or PostgreSQL . The advantage of PostgreSQL over MySQL would be the degree of standardization, MySQL Spatial is designed to match the openGIS standards, but PostgreSQL is openGIS compliant, it has been validated and certified by the OGC . [28] [31] [31]

The PostGIS extension adds new data types such as geometry shapes, geography information or rasters for instance. It also add some functions, operators and index improvements that enhance the use of those types. [5]

The use of a relational database will allow us to easily insert new data in a map, without having to handle complicated queries for simple actions. If we need to add a wall, we simply create a new tuple in the 'Line' table

of the database, with the appropriate information for each points of the line forming the wall, instead of re-creating a new map and re-inserting every items in the database tables.

In this way, we would avoid unnecessary computation process, and we could handle more data.

3.2.3 New Hierarchy

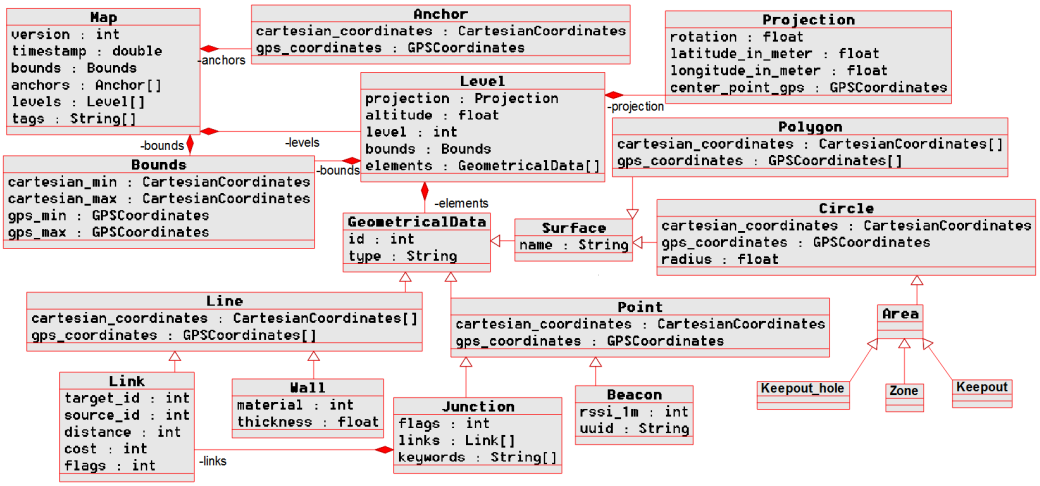

Figure 3.3: New Hierarchy of the Map

As described in the Figure 3.2.3, the hierarchy of the data has been modified: The different elements of a level are now a set of geometrical data, that is an abstract data type implemented as points, lines or surfaces. A surface might be a polygon that contains a set of coordinates, but could also be a circle with a unique coordinate and a radius to define its size. The anchor concept was added, which are the reference coordinates that will be used with the projection to convert the Cartesian coordinates in GPS coordinates. The map components was also modified, it includes a timestamp to support the history possibility in the project. The bounds of a map, or of a level, now contains four coordinates containing the minimum and maximum coordinates, in a Cartesian and in the GPS perspectives.

This organization keeps the general composition and the logic of a map, but sets every specific objects in an abstract generic type of data. In this way, a Link is not a proper item itself, but is a Line with some additional data, a Wall is also a line but with different additional data. This allows us to store all specific types in only one table of the database instead of having one

table per specific type. Abstraction allows us to reduce the code to write, by merging common parts together, it makes it easier to understand the project, and to extend the project with new features or new objects.

3.2.4 Database Design

The handling of the database will be made by an Object Relational Mapping tool (ORM), it is an abstraction layer between the server and the database, which is a tool used to convert object oriented data to a relational oriented data that can be stored in a relational database. This technique eases the handling of the database, by automating the correspondence between the database and the objects of the used language. [39]

The chosen tool will be detailed in the following part 3.3.

As the chosen solution is a relational database, some changes will need to be done in order to store the maps organized as in the Figure 3.2.3. Each geometrical data should have a history, so the abstract type Geometrical Data will include a timestamp in order to record every location it could have been placed.

PostgreSQL includes the basic SQL data types, and its extension PostGIS adds support to have native basic geometrical data types. PostGIS do not natively include round areas data types, but this kind of shape can be easily represented as a unique point with a radius. The database will be used mostly for storing the data, and the computations will be handled by the server, so not having native types for each used shapes is not a problem. But, using an ORM prevents us from using the specific PostGIS data types, so we cannot use the Point object, therefor it will be represented as an array of two floating point values. As a line or a polygon is an array of Points, those types will be represented as arrays of arrays containing two floating point values.

Generally, the GPS coordinates are not stored in the database, because they will be processed and calculated by the server from the Cartesiancoordinates, when a map is retrieved. This principle applies for every gps coordinates, except for the different anchor points, that will be used to calculate all gps coordinates and the projection values.

3.2.4.1 Maps

The maps stored in the database contain an identifier for other objects related to each maps, it includes a version number and a timestamp to have a history of the different states. Finally the maps can include different tags stored in an array of text.

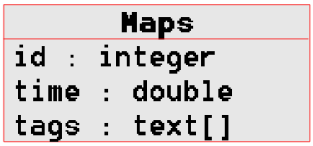

Figure 3.4: Table containing the maps

The identifier, version number and time value cannot be null.

3.2.4.2 Levels

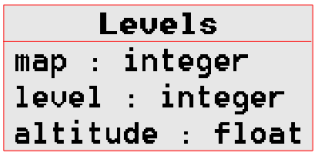

Figure 3.5: Table containing the levels

A level is related to a map, so it includes its identifier. The level is identified by its map identifier and the floor it represents, named level in the table. It also contains an altitude value.

The map and level identifier cannot contain null values.

3.2.4.3 Bounds

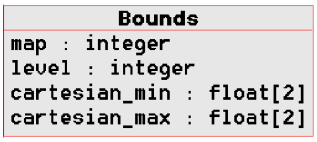

Figure 3.6: Table containing the bounds of the maps and the levels

The bounds of a map or of a level contains the minimum and maximum values of Cartesianand gps coordinates, and the identifier of the related map or the related level.

A value with no level identifier is related to a map only, and bounds with a map identifier and a level identifier is related to the level of a map. The only field of this table that can have a null value is the level identifier.

3.2.4.4 Anchors

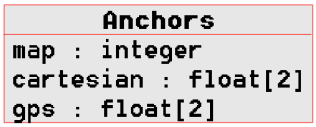

Figure 3.7: Table containing the anchors of a map

The anchors of a map is linking Cartesiancoordinates to gps coordinates. It is related to a map. There cannot be any null value in this table.

3.2.4.5 Polygons

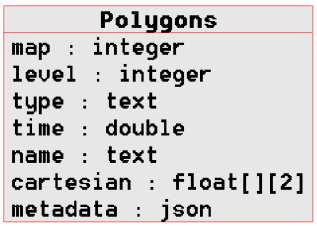

Figure 3.8: Table containing the polygons of a level

A polygon is related to the level of a map, those fields cannot be null. It can include a type and a name, those fields are optional. A timestamp value is used for the history of the polygon. Finally it includes two arrays of the cartesian and gps points that compose the polygon. It can optionally include additional metadata as a JSON value. The only null possible values are the metadata, the name, and the type.

3.2.4.6 Circles

A circle is linked to the level of a map, includes a timestamp value and can include a type, a name and metadata values. It necessarily include a central coordinate, and a radius.

3.2.4.7 Lines

The lines of a level include the non optional fields such as the map identifier, the level identifier, the timestamp value, a set of coordinates. It can optionally include a type value, and metadata.

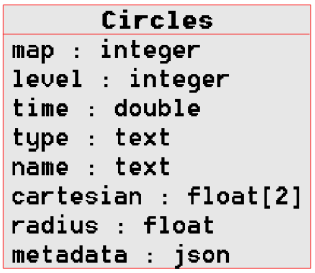

Figure 3.9: Table containing the circles of a level

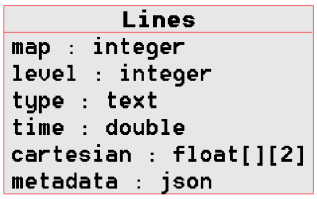

Figure 3.10: Table containing the lines of a level

3.2.4.8 Positions

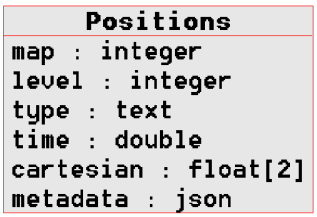

Figure 3.11: Table containing the single points of a level

The different points of a map include the mandatory fields of the map and level identifiers, a timestamp value and its coordinates. It can include the optional fields of type of position and metadata.

3.3 Software Stack

In order to provide a solution easy to use, with as few requirements as possible, we will use the different web technologies to develop the application.

This will allow anyone to use it without having to install other software than a web browser, which is nowadays included in every operating systems.

The entire web service is including the server part and the client part. The server component is the interface between the clients and the database, when a client will save a map for instance, it will first send the data to store to the server, then the server will organize the data in the database. When a client will request a map, the server will read the database and return the map to the client.

On the other side, the client is the component that handles the map creation, and the request of an existing map. As the entire project is meant to include collaborative work, the clients and the server will need to synchronize with each others, so that two clients can create or edit a map at the same moment.

NodeJS is a recent technology that builds client server Javascript applications, as a new web technology, it has been created in order to provide highly scalable solutions. It also includes a Push capability, this will be useful to have a collaborative application, every changes on the map will be quickly made on all the clients editing the same map. For those reasons, the web application will be implemented using the NodeJS technology, including some frameworks, modules and libraries. [10] [37]

Twitter Bootstrap is a collection of tools that help the creation of a web page, it includes HTML and CSS codes for adding forms, buttons, navigation tools and other interactive elements. This framework will be useful to have a faster development of the user interface part. [12]

GruntJS is a JavaScript task runner, it will allow us to automatize the repetitive tasks such as minification of the code or running the tests of the applications for instance. The plugins catalog is quite big, and can allow us to automate a lot of different tasks quickly and effortlessly. [3]

PaperJS is a graphical library for JavaScript, it can be used to create vector drawings. It is a free MIT licensed tool that will be used to draw the maps on each clients using the application. It includes powerful features such as drawing basic vector graphics, but also curves, and more complex elements. [4]

Yeoman consists in tools and frameworks that will help the development process, by quickly building the applications, generating templates, handling the dependencies. The work flow comprises three general tools:

Yo A scaffolding tool that is used to generate the project basis and to generate the configuration files such as build system files, the dependencies of the project file.

Grunt or Gulp The build system tool that is used to build, preview and

test the project.

Bower or npm The package manager, used to automatically retrieve dependencies of the project, Bower and npm are two famous solutions, and can be used together as well.

[7]

ExpressJS is a small web application framework that includes basic tools to handle routes in an easier way. It can also be used to include templates in the application. [2]

AngularJS is an open source JavaScript framework, it is extending the HTML language, including new tags, databinding between the HTML view and the JavaScript controller.

Sequelize is a popular Object Relational Mapping (ORM) library, it is developed in JavaScript, and can be used in a NodeJS environment. It allows the mapping of JavaScripts objects to the relational databases such as MySQL , MariaDB , PostgreSQL and SQLite . This library will allow us to avoid handling the database, and to change from PostgreSQL to another one if it is required. The development of the whole project now exclude the conception and designing of a database.

3.3.1 Collaboration tool

3.3.1.1 Meteor

Meteor is a complete JavaScript framework. It introduces new paradigms concerning the client-server architecture, and allow the development of the server and the client with the same language and the same API. This choice make it easy to transfer a process from the server to a client, or the other way around. Meteor makes it easier to develop collaborative applications, it includes features used to have responsive solutions, with modern interface. It is published as an MIT licensed software. [36]

3.3.1.2 DerbyJS

DerbyJS is a full stack framework. It is designed to help the developer to implement real time collaboration applications, and to make it easier to develop components for the user interface, through bindings between the views and the models for automatic updates in the view when its attached data changes. It is published as an MIT licensed software. [1] [26]

3.3.1.3 TogetherJS

TogetherJS is a JavaScript library developed by Mozilla that helps with real time collaborative work. With this library, each users will see the cursors and clicks of the other clients, they will see what has been edited, what is being watched, the scroll position. The forms of a web page can be filled by two clients. It is a customizable tool, its aspects and behavior can be edited or extended depending on the needs. However, TogetherJS does not give real time persistence, and it is up to the developer to implement it. TogetherJS only synchronizes the sessions in the browser. The library is published as a Mozilla Public Licensed v.2.0 software, which is a permissive free license. [6]

3.3.1.4 Final Choice

The criteria for choosing a solution rather than another are the following:

Free license As it is an important focus of the project, and we want to be able to redistribute some parts of it as free and open source software, the tool used should be easy to use and to redistribute.

Restricted solution We want to handle collaborative data, we do not want to include unnecessary features possibly already included in the other libraries and frameworks.

Limited features The current focus concerning collaboration in the project is to allow one user to be monitored by one or several others, so in the first place we do not need complicated features.

	License	**Size**	**Features**
Meteor	MIT	Full stack framework	Collaboration Interfaces APIs
DerbyJS	MIT	Full stack framework	Collaboration Databinding Routing
TogetherJS	MPL	Small library	Collaborative

Table 3.1: Comparison of popular collaborative JavaScript tools, libraries and frameworks

Among the compared solutions, all of them are distributed as open source software. Meteor and DerbyJS are both full stack framework which means that both include other libraries, that might not be relevant for the project, or even solutions for problems that are already handled by other libraries. On the other side, TogetherJS is specialized in collaborative features. Concerning the included features, TogetherJS only includes collaborative elements, whereas Meteor and DerbyJS also include Interfaces, APIs, databinding or

Routing. Incidentally, those features are already handled by AngularJS, NodeJS itself.

Considering the previous analysis, we will use the library, developed by Mozilla, TogetherJS, to handle collaboration.

4 Implementation

In this chapter, we will describe how the technical project is working, and how it has been organized. We will focus on some important parts of the project, that one may consider the most relevant:

Actions The principle of the actions relative to the map data manipulation.

Application Programmer Interfaces The different use cases of the API.

Client side The client organization, and data handling.

Collaborative features The features included that makes the project collaborative.

4.1 Actions

This section will describe our actions paradigm, we will first explain its principle and then we will list and describe every possible actions.

4.1.1 Principle

A perspective previously discussed was the benefit of sending as little data as possible, whether it is between the programming interfaces and a client, or between two collaborative clients. As it is not useful to send every existing data, it will be restricted to the minimum required and a collaborative message will only contain the most essential part of the data.

To achieve this, we will implement a solution that approximates the Command design pattern: For each action accomplished on the map, a new "Action" object will be created, and represented as a .JSON object.

As described in the figure 4.1.1, the idea of our actions principle is that when the main user is manipulating data on the map, an action object is created, containing the required information for applying this manipulation.

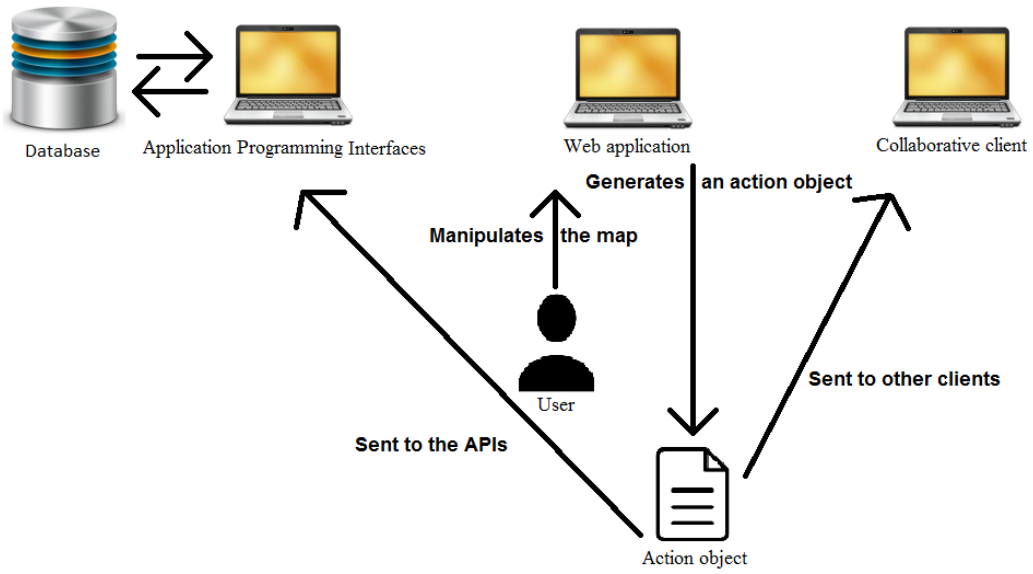

Figure 4.1: Actions concept principle scheme

This object is readable by the clients and the application programming interfaces. So when something is changed in the map of the user, it is quickly replicated in the database through the application programming interfaces and in the maps of the monitoring users.

4.1.2 Tasks

As shown in the table 4.1, every item needs to be created or deleted. No object requires a modification feature, because in the first place modifying an object can be handled by creating a new object with new values, and deleting the old object.

The first focus will be to implement the “CREATE“ and the “DELETE“ actions, and to take care of the “MODIFY“ actions later, in the first time any edition of an object would be made by deleting it and re-creating a new one. The modification actions will be implemented later, depending on the needs of the application.

A special kind of action also exist to spread to the other clients the changing of level in the map editor. The action is name “SETLEVEL“ and simply contains the floor level of the level that has been set by a client. As this action is entirely used by the editor itself, it is not sent to the database.

	CREATE	MODIFY	DELETE
Map	✓	✗	✓
Anchor	✓	✗	✓
Bounds	✓	✗	✓
Level	✓	✗	✓
Circle	✓	✗	✓
Surface	✓	✗	✓
Junction	✓	✗	✓
Point	✓	✗	✓
Line	✓	✗	✓
Polygon	✓	✗	✓

Table 4.1: Listing of all required actions per items.

4.1.3 Structure

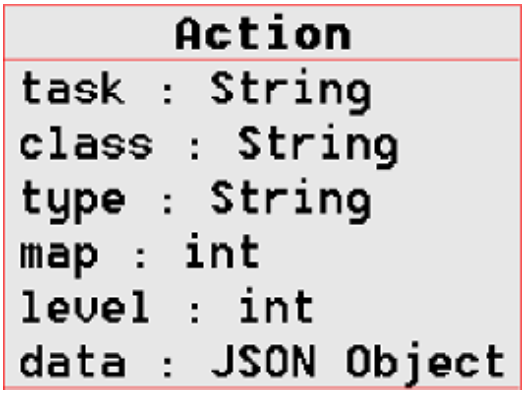

Figure 4.2: Structure of an action

As shown by the figure 4.1.3 describing an action “CREATE“, “MODIFY“ or “DELETE“, the action object contains an attribute task, containing the value “CREATE“, “MODIFY“ or “DELETE“, the identifiers of the map and the level of this map. It also includes the class of item that is being processed, Bounds, Anchor, Position, Line, Polygon or Circle, and it’s subclass described in the attribute type. Finally it includes an attribute data, that contains the data required to describe the item, its position, its different points, or radius for instance.

4.2 Application Programming Interfaces

4.2.1 Operations

The programming interfaces need to handle every existing actions previously discussed, but it will also be used for retrieving data required for building the maps in the client or to generate the .NDD files.

4.2.2 Map retrieving

The clients need to retrieve the maps in the current status, to do so, they will use an API that retrieves all data of the selected map.

It is done when a client requests the map editor application, with a given map identifier in the parameters, the API will retrieve every item composing the map, transform them in a list of "CREATE" actions to be processed by the clients, in order to recreate the map in its current status.

4.3 Client side

The client side is the part used by the final user to create and modify its map. It is a web based solution, usable on any browser of any platform.

4.3.1 Initialization

As described in the figure 4.3.1, when a client in initialized, it first calls the API for getting the map data, the API retrieves the data from the database and returns an organized JSON data representing the map. The client then processes the JSON data and converts it into a PaperJS project, which is displayed in the web application.

4.3.2 Manipulation

For each data manipulation in the client application, the map should be changed. The figure 4.3.2 shows that, when a client is editing its map, it modifies the PaperJS project, but does not change anything in the original JSON data containing the retrieved map. Once those changes are applied an action object is generated containing the required information to perform the changes replication in the database and for the potential collaborative clients.

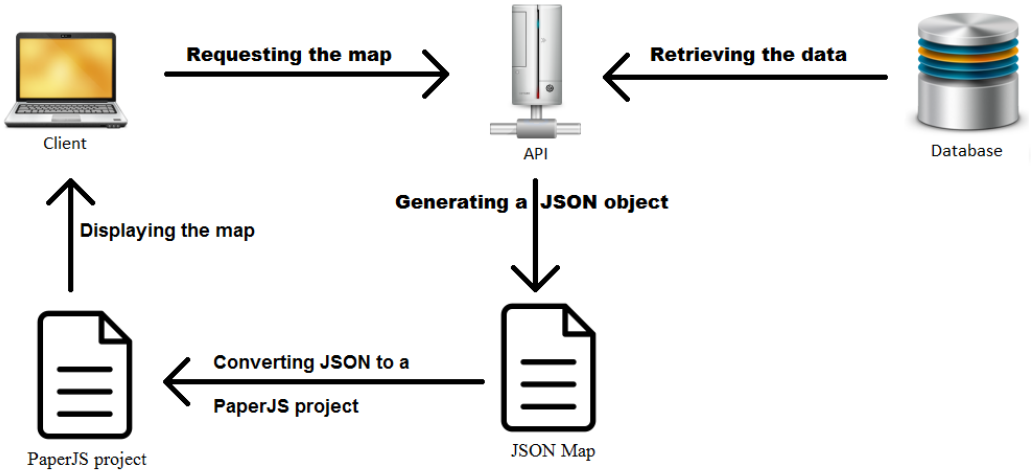

Figure 4.3: Initialization of the client

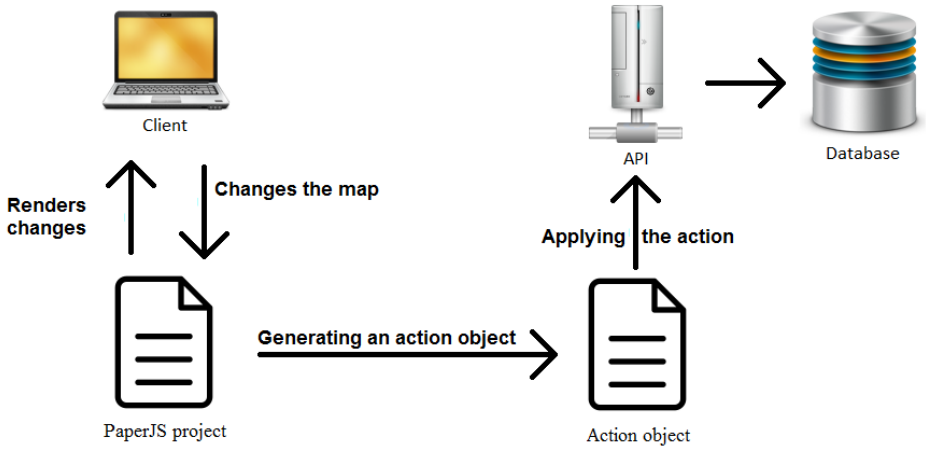

Figure 4.4: Data manipulation from the client

4.3.3 Undo redo

Any activity on a map can be represented as an Action object, and can be spread to the other clients and databases. The current state of the project only includes 'CREATE' and 'DELETE' actions, and cancelling and action simply means that we have to apply the opposite action.

To cancel the 'CREATE' action of an item, we just need to create a 'DELETE' action with the similar data. And to cancel a 'DELETE' action

of an existing item, we simply have to create a 'CREATE' action with the same data.

In other words, the undo feature would be to copy the last processed action, to change the action value of the object: 'CREATE' becomes 'DELETE', and 'DELETE' will become 'CREATE'.

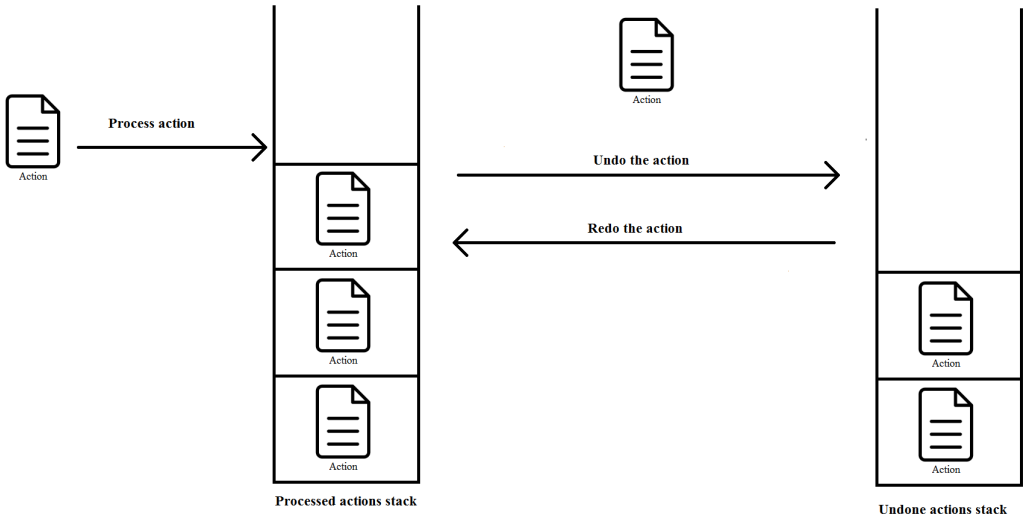

Figure 4.5: Undo redo feature functioning

To undo actions, we need to store the processed actions, and to redo actions, we also need to store the undone actions. Basically this would work with two stacks: As described by the figure 4.3.3, one stack containing the processed actions, each time an action is being processed, we store it in this first stack. When we need to undo an action, we simply pop the first element, we create a copy of the action, swap its 'CREATE' or 'DELETE' value, process it, and then we put the original unswapped value action in a second stack containing the undone actions.

So when we need to redo a previously undone action, we simply pop the undone actions stack, process it and finally push it back in the first stack.

4.4 Collaborative features

As discussed previously, the first focus of collaborative work will be about monitoring a user editing a map, without handling any co-editing possibility: simply checking the different operations of the main user, mainly by replicating its actions to the collaborative users.

4.4.1 Cursor and clicks sharing

The used collaborative JavaScript library, TogetherJS, includes as a first basic feature, the sharing of cursor positions and clicking information for all clients connected to the same page. This allows a real-time monitoring on what elements the main user is editing. But this does not include the interface of the other clients, so when an edition is made, it needs to be manually replicated through the database or through direct replication between the clients.

4.4.2 Actions sharing

TogetherJS makes it easy to send messages between two clients using the same web page, so whenever the main user will finish its editing, an action object will be created, it will be send to the programming interfaces, but also to the collaborative users, that will process it so that the edition can be displayed in their own interfaces.

5 Validation

5.1 Description

This chapter will describe the testing system of the application implemented in the previous chapter.

Software testing is an important component and part of software development, and it is required to validate that a program is correctly running. Four Different layers of testing exist:

Unit testing is the lowest layer of testing, it is a method that will independently test different units of a program to check that they correctly work individually. The first step of a unit test is to define the specification, defining the predicted output of the test with a known input. Then once the test has been run, if the predicted output is correct, we can assume that the test has been successfully passed, and that the test unit works as it should.

Integration testing is the step following unit testing, after the developers have validated their work and their fixes, they regroup their modifications. The point is to define a new version based on a maintenance or a development version. It usually includes revision control software. Nowadays, integration testing has been replaced by continuous integration that includes unit testing and integration testing. The aim of integration testing is to validate the functioning of bigger parts of the program, no simply focusing the test on a single function, but verifying that a whole feature correctly works. It can be done as **Bottom Up Testing**, which means that the low level components will be tested first, and then the highest level components, it can also be done the other way around as **Top Down Testing**, where higher components will be tested before lower level components, or it can finally be done as a mix between the two as **Sandwich Testing**.

System testing is the process that will check if the system can correctly run the developed program or application. The tests take as input all the integrated components that sere validated by the previous testing methods. It is not required to know the content of the software components, its code, or its logic to run the system testing. It includes different kinds of testings such as: usability, performance, compatibility, stress, security, aptitude, maintenance, installation, smoke, scalability, graphical user interface, load, volume testings ...

Acceptance testing is a process that aims to verify that the developed product matches the requirements of the initial specifications. It is the verification that the developed product is what the customer or the end user really needed.

[29]

As said previously, testing is a major part of development, therefore the implemented project requires to be tested.

The web application is mostly visually tested along the development, but as the different items can be displayed separatly between the application and the database, it is required to test the action processing on the server side.

A specific application programming interface has been developed to verify the good functioning of the data storage, and will be described in the following section 5.2.

5.2 Testing System

A unit testing system has been designed to verify the good functioning of the interactions between the actions to be processed and their result on the database. It is a simple application programming interface that takes a list of files as an input, that are present on the server side, and will process each one of these files one after the other.

The initial testing files are essentially designed to test basic insertion and deletion of an item in the database, by processing an action, but as each files are completely independent, it is easy to customize the tests and to have more complex tests, to extensively check different use cases.

As described by the figure 5.2, the application programming interface takes a list of files to process the tests. Each file processes one or several actions and some interactions with the database to check if the wanted tests have been correctly made. And finally the application programming interface returns a list of results containing booleans to describe if each unit correctly works.

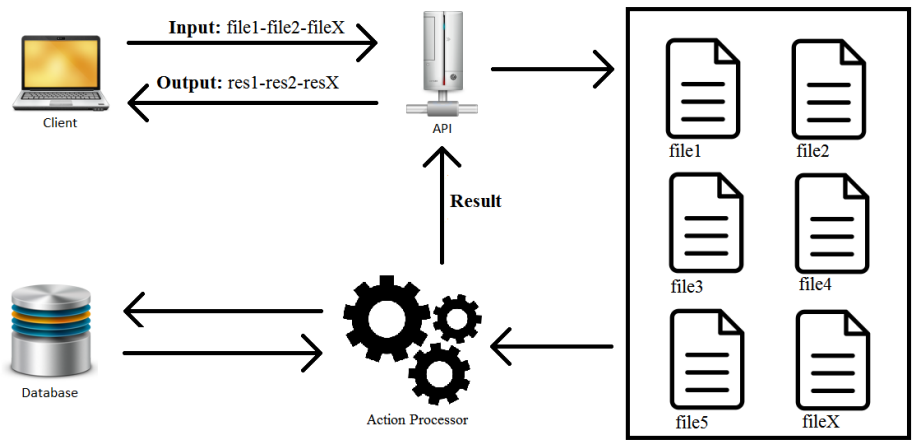

Figure 5.1: Testing system general overview

The initial test files are simply verifying the creation and deletion of generic items in the database:

Level This test simply tries the creation of a level with a given background and a given altitude, and tries to delete it afterward.

Anchor The script for anchor testing processes a creation and a deletion of an anchor on the virtual level created before the starting of the tests.

Bounds The bounds testing first define the bounds of a map, verifies that it worked, and as there can only be one bounds item defined per map, it checks if the creation of new bounds successfully destroys the previous bounds, and finally delete the current bounds.

Circle Line Polygon Position Those four tests simply process the creation of a new item, verifies that it has been successfully created in the database, then processes the DELETE action of the same item, verifies that the database doesn't contain it anymore, and returns the result to the user.

5.3 Results

To process the tests, we will call the application programming interface with the input containing all files that were quoted in the previous section5.2, the successful tests are displayed with the green color, and the failed tests are displayed with the red color:

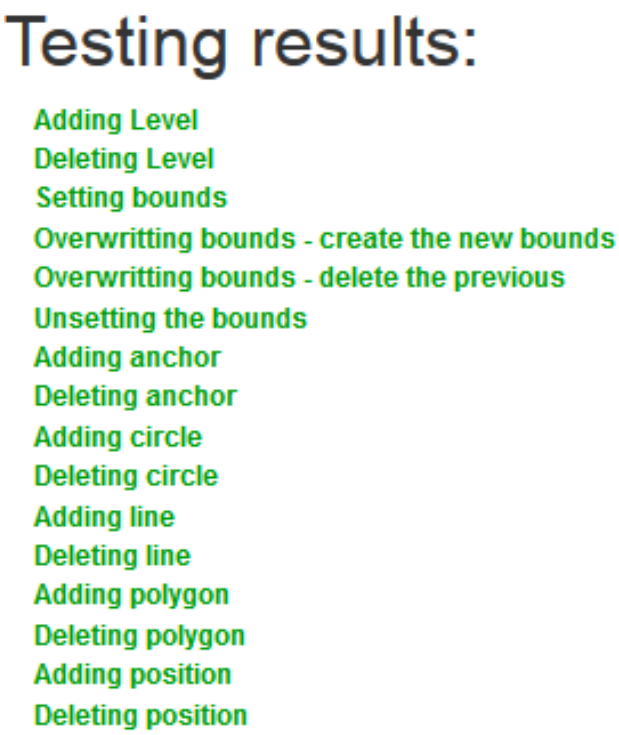

Figure 5.2: Testing results

5.4 Conclusion

The first basic unit tests have been designed, written and successfully passed, but eventually, more complex testing scripts will be written to test more complex sequences, with different items that might interact and interfere with each other.

6 Usability Testing

Concerning the implementation, an essential matter was to provide a solution easy to use, as CAD software are complicated to understand, require training and practice to be mastered, the developed solution had to be accessible for anyone.

After the implementation of the minimum valuable product, the web application has been submitted to usability testing, and experimented by inexperienced users, to evaluate its accessibility. In this way, the feedback will be used to improve the application, and to make it even more easy to use.

Usability testing gives the command of the program to the future user, it draws a picture on how the user will use the application, how it will react [30].

The following sections will describe the user interface itself, then there will be the content of the survey that the experimental users had to answer, after that, the results of the survey will be presented, a feedback from CAD software experienced users will also be given. Finally an analysis of the previous results will be given, to understand what kind of evolution the application needs.

6.1 User Interface

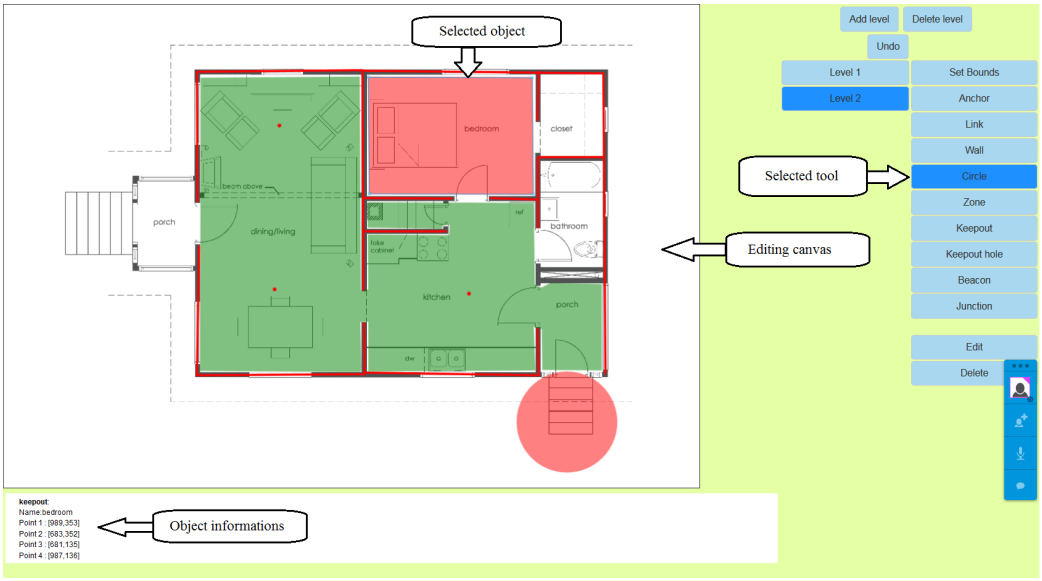

Figure 6.1: Graphical user interface

As shown in the figure 6.1, the graphical user interface is divided in three parts, the upper left part contains the editing canvas, this is the part where the map is displayed and edited.

The upper right part contains the controlling buttons:

Level control buttons used to create new levels, or to delete the currently selected level.

Navigation buttons used to navigate between the different levels. The button of the currently selected level has a darker color in order to easily see which level is being displayed.

Tool buttons used to create the different objects of the map: bounds, anchors, circles, areas, lines.... This section also contains the editing button used to edit the metadata of the different objects created, and the delete button that is used to remove those objects.

Finally, the lower part of the interface contains a displaying zone, that is used to detail the data of the selected item. Depending on the object, it will display its core data and its metadata.

Depending on the tool button used, some additional information can be required: a popup window will be displayed on top of the main interface and the user will be able to fill the required fields.

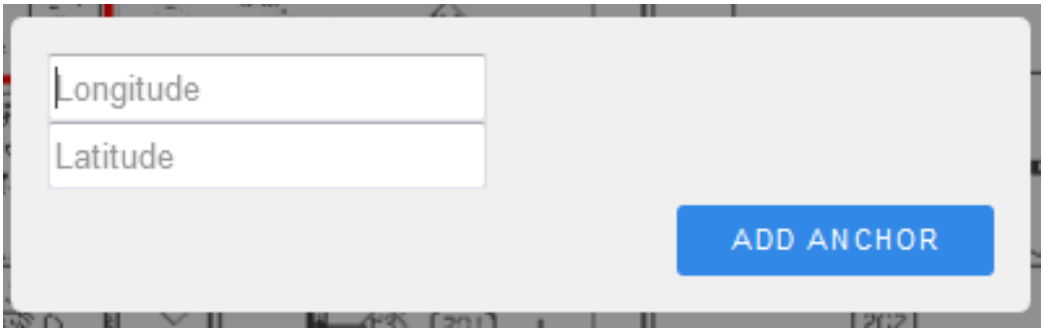

Figure 6.2: Popup window used to set the GPS coordinates of an anchor.

The figure 6.1 shows that when an anchor is created, it is required to define its GPS coordinates, in order to bind the GPS coordinates to the Cartesian coordinates of the map. The popup window requires a latitude and a longitude.

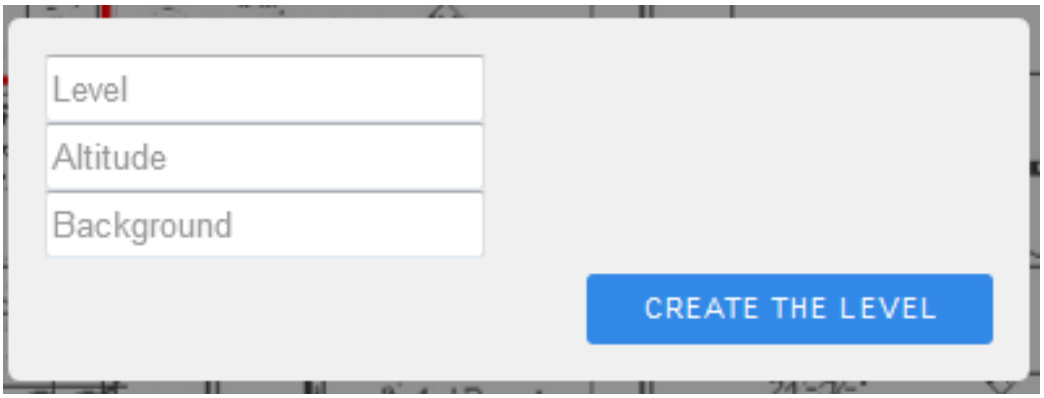

Figure 6.3: Popup window used to set the values of a level.

The figure 6.1 shows that the creation of a new level in the map requires some data, the floor level is mandatory. It can also be useful to set an altitude, and a background image (typically, the floor plan of the building level that has to be digitized).

As displayed in the figure 6.1, the different existing areas (zone, keepout or keepout_hole), currently require a name.

Figure 6.4: Popup window used to set the metadata of an area

Figure 6.5: Popup window used to set the metadata of a beacon

As shown by the figure 6.1, in order to have functioning beacons, each one of them requires a unique identifier, an altitude, and a Received Signal Strength Indication value.

Figure 6.6: Popup window used to set the metadata of a junction

The popup window for junction editing, showed in figure 6.1, requires a name and a unique identifier.

6.2 Survey

The survey used to grade the application usability consists in a series of steps to process on the application, for each steps the user will grade its difficulty from 0 to 5, 0 meaning that the task is complicated to process, 5 meaning that the operation is easy to handle.

- Creating a level
- Moving / Zooming
- Defining the bounds of a level
- Creating areas (zones, keepouts, keepout_holes)
- Creating circles
- Creating anchors
- Creating walls
- Creating beacons
- Deleting elements
- Optional comment

6.3 Results

6.3.1 Inexperienced users

Operation	Average value
Creating a level	2,5
Defining the bounds of a level	3,5
Creating areas (zones, keepouts, keepout_holes)	3
Creating circles	4,5
Creating anchors	4,5
Creating walls	4,5
Creating beacons	4,25
Editing areas (zones, keepouts, keepout_holes)	2,5
Deleting elements	3,25
Moving / Zooming	4,75

Table 6.1: Existing geospatial databases

6.3.2 Feedback

"About the bounds, I didn't get it at the first time. More explanations are welcome !"

– Florian Duros

"Easy to use, but lack of intuitivity"

– Anonymous tester

"Reclick for validation. What are the anchors ? What are the beacons ?"

– Anne Busnel

"Some features are simple to use, but some are not very intuitive, for instance, Delete function should allow user to select an item first. Also creating areas was easy but a bit confusing. There should be a visual representation of the elements used for drawing, like in a graphical editor, there could be icons for line circle etc.; and after completing the drawing, one should be able to mark the figure as a certain area (Zone, Keepout etc.). These specific areas

could be checkboxes, not buttons as Delete or Edit. In general, functions could be visually represented in separate groups, which would improve the usability of the tool."

– Jelena Pantovic

6.3.3 Experienced user feedback

Zooming and panning together with working with the tools was something I expected to be possible. Finishing a zone by re-selecting the same tool was very different from other UIs I've used in the past. Adding beacon meta data required a second click with edit, while adding an anchor did not. Adding anchors was easier than adding beacons. Deleting was very easy. It was nice that undo and redo were supported, it gave confidence when working with the management system.

– Mikko Virkkilä

6.4 Analysis

Concerning the Inexperienced user testing, the results show that the different tools are quite easy to use, but there is a lack of intuitivity on how to use the different tools, some tool buttons deactivate themselves after the creation of an item and some have to be manually deactivated.

A standardization of the behavior of the tool buttons would help the user not to feel lost. The creation of circles, single position items and lines have almost unanimously been reported as easy to use and intuitive, the behaviour of the unintuitive tools will be modified and rethought in order to approach the using method of the intuitive ones.

A recurring problem was also the fact that the users did not know the purpose of some kind of objects such as anchors or that the attribute called "level" of a level represents the floor number, and therefore has to be a numerical value.

The features of moving and zooming on the map is globally satisfying, however improving it by combining the two operations, instead of simply zooming on the center point, moving the center point closer to the cursor and then zooming to this new center point.

In terms of user interface, studying the different CAD software could be interesting as well in order to have an overview of the existing interfaces,

and their logic. The behavior of editing the metadata of the different objects is not the same for every items, and appears to make it harder to use. A standardization of this behavior could improve the usage and avoid the user to be confused.

To conclude with the general usability of the implemented solution, the user interface is not intuitive, the behaviors are not standardized and therefore confuses the users.

7 Conclusion and Future Work

7.1 Discussion

7.1.1 Abstraction

Scientific questions were discussed, reducing the required objects to its minimum, having generic types of objects and having a solution as much abstracted as possible.

> *"The essence of abstractions is preserving information that is relevant in a given context, and forgetting information that is irrelevant in that context."*
>
> – John V. Guttag [17]

Using abstraction will make future changes and code extensions easier to do.

However, looking back on some parts, the Anchor object seems to be a useless type of object as it only includes a single point position, and could easily be represented as a Position object, and its data (latitude and longitude values), could be stored as metadata. Same problem for the other "specialized" items such as Bounds or Projections, having one table in the database fully dedicated to this kind of object seems redundant and unnecessary, and could easily be converted as generic items as Position, Line or Polygon.

A second deeper inspection of the project could improve the abstraction aspect.

7.1.2 Collaborativity

The collaborative part of the application was visually tested, even if it was one of the main feature for this research, deploying a distributed testing solution was not an option in the given time.

Considering the functioning of the collaborative parts of the solution, the principle of the Action objects demonstrated themselves to be a efficient way of replicating changes on a project, it makes it easy to perform one action on different actors of a system. However, the library used to handle the collaboration between two clients does not seem to be a good technical choice. The loading of the library does not work correctly all the time, it seems to provoke some slowings on the application, and the big amount of useless messages in the browser log file makes it harder to debug the application.

Even if the library is working correctly and was efficient enough to develop a minimum valuable product, it would be preferable to either use another library such as MeteorJS or DerbyJS, or maybe developing our own small service that would handle the replication of Actions, this could be easily done with one more application programming interface in the solution.

7.1.3 User Interface and User Experience

As seen in the chapter 6, the implemented application is easy to use for some features, some are harder to understand and to use. However, the application has a general lack of intuitivity and needs an improvement of the user experience to really achieve the goal of having a software easily accessible for untrained users, and to replace CAD software.

7.1.4 Technical Choices

Using NodeJS to develop the web application made is easy to set up a web server. NodeJS includes a big amount of extensions, for various purposes, so whenever we want to implement a new feature, and this feature can be solved or handled by an existing NodeJS extension, we just need to retrieve it thanks to Node Package Manager, and to include it in the project.

Using PostgreSQL as the database management system revealed itself a good solution, as it was easy to use generic geometric objects such as lines, points, or polygons and to store their values as native arrays of points for instance. Abstracting the usage of the database with Sequelize was easy to do, and it avoided wasting time on manually handling CRUD (Create, Read, Update, Delete) operations on the database objects.

7.2 Future work

Some fixes of the implementation itself are required, some graphic bugs and functioning incompatibilities discovered during the usability testings.

As described by the previous sections and chapters, the user interface will need to be highly improved in order to provide a better user experience, more intuitivity and more ease of use to the users. Some explanation on what the user is currently doing would greatly help and avoid confusion.

The collaboration library revealed itself working but not perfectly adapted to the solution, going for another library would be a plausible solution, but would probably require to import a big framework in the current solution that would not be specialized for collaborativity between several clients. The best way to solve this, would be to develop a specific solution to handle the Action messages between the clients, possibly with an API that would receive every messages, and that would then regularly be called by all clients to retrieve the different actions.

Object	**Generic Type**
Anchors	Position
Bounds	Position
Projection	Position

Table 7.1: Abstraction of objects

Finally, some types of objects will be removed such as Anchors, Bounds and Projection, and will be converted as generic geometric types, in order to spare unnecessary database tables.

Bibliography

[1] Derbyjs. `http://derbyjs.com/`. Accessed: 2015-02-11.

[2] Expressjs. `http://expressjs.com/starter/faq.html`. Accessed: 2015-02-10.

[3] Grunt the javascript task runner. `http://gruntjs.com/`. Accessed: 2015-02-09.

[4] Paper.js. `http://paperjs.org/about/`. Accessed: 2015-02-10.

[5] Postgis features. `http://postgis.net/features`. Accessed: 2015-02-12.

[6] Togetherjs technology overview. `https://togetherjs.com/docs/#technology-overview`. Accessed: 2015-02-10.

[7] Yeoman. `http://yeoman.io/`. Accessed: 2015-02-10.

[8] Azhar, S. Building information modeling (bim): Trends, benefits, risks, and challenges for the aec industry. *Leadership and Management in Engineering 11*, 3 (2011), 241–252.

[9] Bahl, P., and Padmanabhan, V. N. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2000), vol. 2, Ieee, pp. 775–784.

[10] Cantelon, M., Holowaychuk, T., Rajlich, N., and Harter, M. *Node. js in Action*. Manning, 2014.

[11] Chrisman, N. R., Cowen, D. J., Fisher, P. F., Goodchild, M. F., and Mark, D. M. Geographic information systems. *Geography in America* (1989), 353–375.

[12] Cochran, D. *Twitter Bootstrap Web Development How-To*. Packt Publishing Ltd, 2012.

[13] Crockford, D. The application/json media type for javascript object notation (json).

[14] Eastman, C., et al. An outline of the building description system. In _ (1974), Carnegie-Mellon Univ., Pittsburgh, PA. Inst. of Physical Planning, p. 23.

[15] Goodchild, M. F. Geographic information systems. *Journal of Retailing 67*, 1 (1991), 3–15.

[16] Güting, R. H., and Schneider, M. *Moving objects databases*. Elsevier, 2005.

[17] Guttag, J. V. *Introduction to Computation and Programming Using Python*. Mit Press, 2013.

[18] Hartley, R. I., and Sturm, P. Triangulation. *Computer vision and image understanding 68*, 2 (1997), 146–157.

[19] Herot, C. F., Carling, R., Friedell, M., and Kramlich, D. A prototype spatial data management system. In *ACM SIGGRAPH Computer Graphics* (1980), vol. 14, ACM, pp. 63–70.

[20] Hightower, J., and Borriello, G. Location systems for ubiquitous computing. *Computer 34*, 8 (2001), 57–66.

[21] Honkavirta, V., Perala, T., Ali-Loytty, S., and Piché, R. A comparative survey of wlan location fingerprinting methods. In *Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on* (2009), IEEE, pp. 243–251.

[22] Kaemarungsi, K., and Krishnamurthy, P. Modeling of indoor positioning systems based on location fingerprinting. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* (2004), vol. 2, IEEE, pp. 1012–1022.

[23] Krishnakumar, A., and Krishnan, P. The theory and practice of signal strength-based location estimation. In *Collaborative Computing: Networking, Applications and Worksharing, 2005 International Conference on* (2005), IEEE, pp. 10–pp.

[24] Krumm, J., and Horvitz, E. Locadio: Inferring motion and location from wi-fi signal strengths. In *MobiQuitous* (2004), pp. 4–13.

[25] Lee, G., Sacks, R., and Eastman, C. M. Specifying parametric building object behavior (bob) for a building information modeling system. *Automation in construction 15*, 6 (2006), 758–776.

[26] Mardan, A. Sails. js, derbyjs, loopback, and other frameworks. In *Pro Express. js*. Springer, 2014, pp. 205–214.

[27] Mirzaei, R. S. Spatio-temporal databases for indoor positioning systems. _ (2005).

[28] Momjian, B. *PostgreSQL: introduction and concepts*, vol. 192. Addison-Wesley New York, 2001.

[29] Myers, G. J., Sandler, C., and Badgett, T. *The art of software testing*. John Wiley & Sons, 2011.

[30] Nielsen, J. *Usability engineering*. Elsevier, 1994.

[31] Ramsey, P., et al. Postgis manual. *Refractions Research* (2005).

[32] Roos, T., Myllymaki, P., and Tirri, H. A statistical modeling approach to location estimation. *Mobile Computing, IEEE Transactions on 1*, 1 (2002), 59–69.

[33] Ruud, V. Augmented reality for indoor navigation.

[34] Samet, H. Applications of spatial data structures. _ (1990).

[35] SARCAR, M., Rao, K. M., and NARAYAN, K. L. *Computer aided design and manufacturing*. PHI Learning Pvt. Ltd., 2008.

[36] Strack, I. *Getting Started with Meteor JavaScript Framework*. Packt Publishing Ltd, 2012.

[37] Tom Hughes-Croucher, M. W. *Node: Up and Running*. O'Reilly Media, 2012.

[38] Varshavsky, A., de Lara, E., Hightower, J., LaMarca, A., and Otsason, V. Gsm indoor localization. *Pervasive and Mobile Computing 3*, 6 (2007), 698–720.

[39] Vasiliev, Y. Object/relational mapping. *Beginning Database-Driven Application Development in Java EE: Using GlassFish* (2008), 223–252.

[40] Yang, Z., Liu, Y., and Li, M. Beyond trilateration: On the localizability of wireless ad-hoc networks. In *INFOCOM 2009, IEEE* (2009), IEEE, pp. 2392–2400.

A List of considered databases

Solution	Native	License	Standard
Boeing's Spatial Query Server	x	Proprietary	
AllegroGraph	x	Proprietary	
IBM DB2 Spatial Extender	IBM DB2	Proprietary	
CartoDB	PostgreSQL	Proprietary	
Microsoft SQL Server	x	Proprietary	
SpaceBase	x	Proprietary	
Neo4j	x	GPL	
MongoDB	x	AGPL	
RavenDB	x	AGPL	
RethinkDB	x	AGPL	
Tarantool	x	BSD	
Geocouch	CouchDB	Apache	
AsterixDB	x	Apache	
Postgres-XL	PostgreSQL	MPL	
GEODAS	x	Proprietary	Compliant
SpatialDB	x	Proprietary	Compliant
Teradata Geospatial	Teradata	Proprietary	Compliant
SmallWorld	x	Proprietary	Compliant
Ingres 10.2	x	GPL	Compliant
NRDB	x	GPL	Compliant
H2GIS	H2	GPL	Standard
Linter SQL Server	x	Proprietary	Standard
SpatiaLite	SQLite	MPL	Compliant
MonetDB/GIS	MonetDB	MPL	Compliant
GeoMesa	Apache Accumulo	Apache	Standard
Oracle Spatial	Oracle	Proprietary	Compliant
MySQL Spatial	MySQL	GPL	Standard
PostGIS	PostgreSQL	GPL	Compliant

Table A.1: Existing geospatial databases

(a) In English

Figure A.1: Aalto logo variants